SciSpot

Paper # XXX

ABSTRACT

In this paper, we...

1 INTRODUCTION

Running parallel scientific applications, such as molecular dynamics (MD) simulations, on low-cost cloud transient resources.

The first "big" idea is that simulations are often bag of parallel tasks.

While there has been some past work that looks at running MPI applications on spot instances, our scope is much broader and considers how complete simulation pipelines can be run at low cost.

Spiel on transient instances. Increasingly popular resource allocation model that is being offered by all cloud providers. Very low cost compared to conventional cloud resources, often by up to 10x. However, can be frequently revoked. Thus failure is a common occurrence, and not a rare-event. This is especially challenging for MPI jobs because of its inability to tolerate failures.

However, our insight is that while protecting a *single* job against revocations can require elaborate checkpointing based approaches, we dont necessarily have to do that if we consider that most simulations are composed of a series of jobs that search over a parameter space, and that what is important is the total running time and cost of this entire series of jobs.

Thus, no single job is "special".

Another aspect of novelty is that past work on transient resources used EC2 pricing information to get failure probabilities. However, this is no longer an accurate method. We perform the first empirical study of google preemptible VMs and their performance and availability for HPC workloads.

Another fundamental question is what is a suitable metric in such cases. Conventionally, it is speedup. In the cloud, it is some combination of cost and running time.

, ,

.

2 BACKGROUND

2.1 Transient Computing

2.2 Parallel Scientific Applications on the Cloud

3 DESIGN

Scientific simulation applications consume a large amount of computational resources, and are often used in the context of *exploratory* research, where a large amount of jobs are run with different simulation parameters. This can be either a *parameter sweep*, where a large number of parameters need to be evaluated, or a *search* over a large parameter space for the “right” set of parameters that yield the desired model behavior.

In this paper, we look at the problem of running scientific simulations on *transient* computing resources in public clouds.

Past work has largely been focused on running parallel jobs (such as MPI) in the cloud. However, considering entire *job-groups* or ensembles of jobs presents new challenges and opportunities in timely, low-cost computation.

Our system,SciSpot, is a unified framework for running large job-groups that result from parameter exploration.

Input and some assumptions: We assume that the job-group consists of $J_1 \dots J_N$, with each job evaluating a model on some parameter. The list of parameters to explore can either be generated apriori (as in the case of parameter sweeps), or be dynamically generated as in the case of a search.

This presents us with many challenges in the cloud-deployment of these jobs.

Cloud providers offer multiple types of instances (VMs), with different hardware configuration (such as number of CPUs and memory size). The price of cloud servers is related to their hardware configuration, but it may not be strictly proportional to the hardware performance. For example, a VM with 32 CPUs may not be 32 times the cost of a single CPU VM.

For parallel and distributed applications, the type of servers selected has large implications on their performance. Consider the case of deploying an application on 8 8-core VMs vs. 16 4-core VMs. In both cases, the total number of CPU cores is the same. However, the larger number of VMs requires more communication between the application tasks, and thus may result in performance degradation.

Thus, when deploying applications on the cloud, one has to mindful of the cost and performance tradeoff. However, in the case of transient servers, the story does not stop here.

In addition to pricing differences, the transient availability of instances *also* differs by type. Because the availability of a transient VM is broadly determined by the overall supply and demand of the instances of that *particular* type, the "preemption rate" of VMs often depends on the type of the instance.

Thus, selecting a transient cloud server involves a complex tradeoff between the cost of servers, their performance, and the preemption-rate.

3.1 Trade-offs in Server Selection

4 IMPLEMENTATION

Central controller.

Cloud APIs for launching the jobs.

Slurm.

5 EVALUATION

The contenders:

(1) Run every job on un-tuned on-demand instance (cost and running time)
(2) Run every job on nanohub/big-red-2 (running and waiting time)
(3) Run on transient, restart every time (cost)
(4) SciSpot with early stopping and job sacrificing

5.1 Preemption likelihood curves

5.2 Searching for the best cloud configuration

5.3 Total cost vs. running time graphs

6 RELATED WORK

6.1 Scientific applications on cloud

A classic survey is [6] [13]

Parameter sweep: [2]

Price optimizations for Scientific workflows in the cloud [4]

6.2 Transiency mitigation

[7] classic work on MPI and Spot. Uses checkpointing. Redundancy, but for what? User specified number of VMs. Does not do instance selection. BCLR for checkpointing.

MOre spot and MPI: [5]. FOcussed on bidding and checkpoint interval. But bidding doesnt matter.

[10] is early work for spot and MPI and

[9] a batch computing service

Heterogenity often used, but not useful in the context of MPI jobs. [8]

Selecting the best instance type, often for data analysis computations [1], and [12], and others like Ernest and Hemingway.

All the past work was on EC2 spot market with gang failures and independent markets [5, 7]. However this assumption has now changed, and failures can happen anytime. Our failure model is more general, and applies to both cases.

6.2.1 Fault-tolerance for MPI. [3] has a discussion of checkpointing frequency which is comprehensive.

Replication is another way [11]

6.2.2 Huge amount of work on bidding in HPC. [?] [?]

REFERENCES

[1] Alipourfard, O., and Yu, M. CherryPick: Adaptively Unearthing the Best Cloud Configurations for Big Data Analytics. 15.

[2] Casanova, H., Legrand, A., Zagorodnov, D., and Berman, F. Heuristics for scheduling parameter sweep applications in grid environments. In *Proceedings 9th Heterogeneous Computing Workshop (HCW 2000) (Cat. No.PR00556)* (May 2000), pp. 349–363.

[3] Dongarra, J., Herault, T., and Robert, Y. Fault tolerance techniques for high-performance computing. 66.

[4] GarÃņ, Y., Monge, D. A., Mateos, C., and GarcÃņa Garino, C. Learning budget assignment policies for autoscaling scientific workflows in the cloud. *Cluster Computing* (Feb. 2019).

[5] Gong, Y., He, B., and Zhou, A. C. Monetary cost optimizations for MPI-based HPC applications on Amazon clouds: checkpoints and replicated execution. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis on - SC '15* (Austin, Texas, 2015), ACM Press, pp. 1–12.

[6] Iosup, A., Ostermann, S., Yigitbasi, M. N., Prodan, R., Fahringer, T., and Epema, D. H. J. Performance Analysis of Cloud Computing Services for Many-Tasks Scientific Computing. *IEEE Transactions on Parallel and Distributed Systems 22*, 6 (June 2011), 931–945.

[7] Marathe, A., Harris, R., Lowenthal, D., De Supinski, B. R., Rountree, B., and Schulz, M. Exploiting redundancy for cost-effective, time-constrained execution of hpc applications on amazon ec2. In *HPDC* (2014), ACM.

[8] Sharma, P., Irwin, D., and Shenoy, P. Portfolio-driven resource management for transient cloud servers. In *Proceedings of ACM Measurement and Analysis of Computer Systems* (June 2017), vol. 1, p. 23.

[9] Subramanya, S., Guo, T., Sharma, P., Irwin, D., and Shenoy, P. SpotOn: A Batch Computing Service for the Spot Market. In *SOCC* (August 2015).

[10] Taifi, M., Shi, J. Y., and Khreishah, A. SpotMPI: A Framework for Auction-Based HPC Computing Using Amazon Spot Instances. In *Algorithms and Architectures for Parallel Processing*, Y. Xiang, A. Cuzzocrea, M. Hobbs, and W. Zhou, Eds., vol. 7017. Springer Berlin Heidelberg, Berlin, Heidelberg, 2011, pp. 109–120.

[11] Walters, J. P., and Chaudhary, V. Replication-Based Fault Tolerance for MPI Applications. *IEEE Transactions on Parallel and Distributed Systems 20*, 7 (July 2009), 997–1010.

[12] Yadwadkar, N. J., Hariharan, B., Gonzalez, J. E., Smith, B., and Katz, R. H. Selecting the *best* VM across multiple public clouds: a data-driven performance modeling approach. In *Proceedings of the 2017 Symposium on Cloud Computing - SoCC '17* (Santa Clara, California, 2017), ACM Press, pp. 452–465.

[13] Zhai, Y., Liu, M., Zhai, J., Ma, X., and Chen, W. Cloud versus in-house cluster: evaluating Amazon cluster compute instances for running MPI applications. In *State of the Practice Reports on - SC '11* (Seattle, Washington, 2011), ACM Press, p. 1.